IN THE CLAIMS:

1. (Currently Amended) A method for operating a data storage system, comprising:

creating a writable virtual disk (vdisk) at a selected time, the writable vdisk referencing changes in data stored in the data storage system after the writable vdisk was created;

maintaining a backing store, the backing store referencing data stored in the data storage system which has not been changed since the writable vdisk was created;

loading blocks of the writable vdisk into a memory, the loaded writable vdisk blocks including a writable vdisk indirect block having a plurality of fields, each writable vdisk indirect block field storing a valid pointer to a writable vdisk data block or an invalid pointer representing one of a plurality of holes, where each hole instructs the data storage system to examine a corresponding virtual block number pointer in the backing store;

loading blocks of the backing store into the memory, the loaded backing store blocks including a backing store indirect block having a plurality of fields, each backing store indirect block field corresponding to a ~~field of the~~ writable vdisk indirect block field, one or more backing store indirect block fields having a pointer to a backing store data block;

searching each field of the writable vdisk indirect block for a hole; and

filling each hole in the ~~writeable~~ writable vdisk by replacing each invalid pointer with a pointer to the backing store data block referenced by the corresponding backing store indirect block field to update the writable vdisk to reference both the data which is unchanged since the writable vdisk was created and the data which has been changed since the writable vdisk was created.

2. (Currently Amended) The method of claim 1, further comprising:

dirtying the backing store data block pointed to by the backing store indirect block to enable write allocation of the dirty backing store data block without altering a data content of the backing store data block.

3. (Currently Amended) The method of claim ~~1~~2, further comprising:

    choosing a new pointer for a newly allocated data block containing an unaltered data content;

    setting bits in block allocation structures for the newly allocated data block; and

    placing the new pointer to the newly allocated data block into the field of the writable vdisk indirect block to replace the hole.

4. (Currently Amended) The method of claim 3 further comprising:

    freeing the dirty <u>backing store</u> data block; and

    writing the newly allocated data block to disk.

5. (Currently Amended) The method of claim 4 further comprising:

    releasing an association of the writable vdisk to the backing store to thereby separate the writable vdisk ~~data~~ blocks from the backing store ~~data~~ blocks.

6. (Original) The method of claim 1 wherein the pointers contained in the writable vdisk indirect block fields and the backing store indirect block fields comprise logical volume block numbers (VBNs).

7. (Currently Amended) The method of claim 1 wherein the invalid pointers contained in the writable vdisk indirect block field~~s~~ comprise<u>s</u> a zero logical volume block number (VBN).

8. (Original) The method of claim 1 wherein the plurality of fields in the writable vdisk indirect block are a writable vdisk level 1 buffer and the plurality of fields in the backing store indirect block are a backing store level 1 buffer.

9. (Currently Amended) An apparatus for operating a computer database, comprising:

2   a writable virtual disk (vdisk) created at a selected time, the writable vdisk

3 referencing changes in data stored in a data storage system after the writable vdisk was

4 created;

5   a backing store, the backing store referencing data stored in the data storage

6 system which has not been changed since the writable vdisk was created;

7   a backdoor message handler that loads blocks of the writable vdisk and backing

8 store from disk into a memory of the storage system;

9   a writable vdisk indirect block in the memory having a plurality of fields, each

10 writable vdisk field storing a valid pointer to a writable vdisk data block or an invalid

11 pointer representing one of a plurality of holes, where each hole instructs the data storage

12 system to examine a corresponding virtual block number pointer in the backing store;

13   a backing store indirect block in the memory having a plurality of fields, each

14 backing store indirect block field corresponding to a ~~field of the~~ writable vdisk indirect

15 block field, each backing store indirect block field having a pointer to a backing store

16 data block;

17   a special loading function that searches each field of the writable vdisk indirect

18 block for one or more fields representing a hole; and

19   a write allocator that fills each hole in the ~~writeable~~ writable vdisk by replacing

20 each invalid pointer with a pointer to the backing store data block referenced by the

21 corresponding backing store indirect block field to update the writable vdisk to reference

22 both the data which is unchanged since the writable vdisk was created and the data which

23 has been changed since the writable vdisk was created.

1   10. (Currently Amended) The apparatus of claim 9 wherein the write allocator further

2 ~~comprises:~~ chooses a new pointer for a newly allocated data block containing unaltered

3 data content, set bits in block allocation structures for the newly allocated data block, and

4 place the new pointer to the newly allocated data block into the field of the writable vdisk

5 indirect block to replace the hole.

11. (Currently Amended) The apparatus of claim 10 wherein the write allocator further frees the ~~dirty~~ backing store data block and writes the newly allocated data block to disk.

12. (Currently Amended) The apparatus of claim 9 wherein the backdoor message handler further loads the blocks of the writable vdisk and the blocks of the backing store during periods of reduced processing activity.

13. (Currently Amended) The apparatus of claim 9 wherein the pointers contained in the writable vdisk indirect block fields and ~~the~~ pointers in backing store indirect block fields comprise logical volume block numbers (VBNs).

14. (Currently Amended) The apparatus of claim 9 wherein the invalid pointer~~s~~ contained in the writable vdisk indirect block field~~s~~ comprise a zero logical volume block number (VBN).

15. (Currently Amended) The apparatus of claim 9 wherein the plurality of fields in the writable vdisk indirect block comprise~~s~~ a writable vdisk level 1 buffer and the plurality of fields in the backing store indirect block comprise~~s~~ a backing store level 1 buffer.

16.-18. (Cancelled).

19. (Currently Amended) A data storage system apparatus, comprising:
    means for creating a writable virtual disk (vdisk) at a selected time, the writable vdisk referencing changes in data stored in the data storage system after the writable vdisk was created;
    means for maintaining a backing store, the backing store referencing data stored in the data storage system which has not been changed since the writable vdisk was created;
    means for loading blocks of the writable vdisk from a disk into a memory, the loaded writable vdisk blocks including a writable vdisk indirect block having a plurality

10   of fields, each <u>writable vdisk </u>field storing a valid pointer to a <u>writable vdisk </u>data block or

11   an invalid pointer representing one of a plurality of holes, where each hole instructs the

12   data storage system to examine a corresponding virtual block number pointer in the

13   backing store;

14   means for loading blocks of the backing store from a disk into the memory, the

15   loaded <u>backing store </u>blocks including a backing store indirect block having a plurality of

16   fields, each backing store indirect block field corresponding to a ~~field of the~~ writable

17   vdisk indirect block <u>field</u>, one or more backing store indirect block fields having a pointer

18   to a <u>backing store </u>data block;

19   means for searching each field of the writable vdisk indirect block for a hole; and

20   means for filling each hole in the writable vdisk by replacing each invalid pointer

21   with a pointer to the <u>backing store </u>data block referenced by the corresponding backing

22   store indirect block field to update the writable vdisk to reference both the data which is

23   unchanged since the writable vdisk was created and the data which has been changed

24   since the writable vdisk was created .

1   20. (Currently Amended) A non-transitory computer readable medium executable

2   program instructions executed by a processor, comprising:

3   program instructions that create a writable virtual disk (vdisk) at a selected time,

4   the writable vdisk referencing changes in data stored in a data storage system after the

5   writable vdisk was created;

6   program instructions that maintain a backing store, the backing store referencing

7   data stored in the data storage system which has not been changed since the writable

8   vdisk was created;

9   program instructions that load blocks of the writable vdisk from a disk into a

10   memory, the loaded <u>writable vdisk </u>blocks including a writable vdisk indirect block

11   having a plurality of fields, each <u>writable vdisk </u>field storing a valid pointer to a <u>writable</u>

12   <u>vdisk </u>data block or an invalid pointer representing one of a plurality of holes, where each

13   hole instructs the data storage system to examine a corresponding virtual block number

14   pointer in the backing store;

15   program instructions that load blocks of the backing store from a disk into the

16 memory, the loaded <u>backing store</u> blocks including a backing store indirect block having

17 a plurality of fields, each backing store indirect block field corresponding to a ~~field of the~~

18 writable vdisk indirect block <u>field</u>, one or more backing store indirect block fields having

19 a pointer to a <u>backing store</u> data block;

20   program instructions that search each field of the writable vdisk indirect block for

21 a hole; and

22   program instructions that fill each hole in the ~~writeable~~ <u>writable</u> vdisk by

23 replacing each invalid pointer with a pointer to the <u>backing store</u> data block referenced by

24 the corresponding backing store indirect block field to update the writable vdisk to

25 reference both the data which is unchanged since the writable vdisk was created and the

26 data which has been changed since the writable vdisk was created.

1 21.-22. (Cancelled).

1 23. (Currently Amended) A method for operating a data storage system, comprising:

2   creating a writable virtual disk (vdisk) at a selected time, the writable vdisk

3 referencing changes in data stored in the data storage system after the writable vdisk was

4 created, the writable vdisk having a plurality of holes where each hole instructs the

5 storage system to examine a corresponding virtual block number pointer in a backing

6 store;

7   maintaining the backing store, the backing store referencing the data stored in the

8 data storage system which has not been changed since the writable vdisk was created;

9   searching each field of the writable vdisk for a hole; and

10   filling each hole in the writable vdisk by replacing each hole in the writable vdisk

11 to point to ~~the~~ <u>a</u> data block referenced by ~~the~~ <u>a</u> corresponding backing store indirect block

12 <u>of the backing store</u> to fill each hole of the ~~writeable~~ <u>writable</u> vdisk with the data block

13 <u>referenced by the corresponding backing store indirect block</u> and thus update the writable

14 vdisk to reference both the data which is unchanged since the writable vdisk was created

15 and the data which has been changed since the writable vdisk was created.

24. (Previously Presented) The method of claim 23, further comprising:

dirtying the data block pointed to by the backing store indirect block to enable write allocation of the dirty data block without altering a data content of the data block.

25. (Currently Amended) The method of claim 23 24 further comprising:

choosing a new pointer for a newly allocated data block containing an unaltered data content;

setting bits in block allocation structures for the newly allocated data block; and

placing the new pointer to the newly allocated data block into the a field of the a writable vdisk indirect block to replace the hole.

26. (Previously Presented) The method of claim 25, further comprising:

freeing the dirty data block; and

writing the newly allocated data block to disk.

27. (Currently Amended) The method of claim 26 further comprising:

releasing an association of the writable vdisk to the backing store to thereby separate the writable vdisk data blocks from the backing store data blocks.

28. (Currently Amended) The method of claim 23, further comprising:

including logical volume block numbers (VBNs) in the pointers contained in the a writable vdisk indirect block fields of the writable vdisk and the a backing store indirect block fields of the backing store.

29. (Currently Amended) The method of claim 23 28, further comprising:

using a zero logical volume block number (VBN) as the an invalid pointers contained in the hole of the writable vdisk indirect block fields.

30. (Currently Amended) The method of claim 23, further comprising:

2   using a writable vdisk level 1 buffer for ~~the~~ a plurality of fields in ~~the~~ a writable

3   vdisk indirect block of the writable vdisk and using a backing store level 1 buffer for ~~the~~

4   a plurality of fields in the backing store indirect block.

1   31. (Currently Amended) A data storage system, comprising:

2   a writable virtual disk (vdisk) created at a selected time, the writable vdisk

3   referencing changes in data stored in the data storage system after the writable vdisk was

4   created, the writable vdisk having a plurality of holes , each hole instructing the storage

5   system to examine a corresponding virtual block number pointer in a backing store;

6   the backing store referencing data stored in the data storage system which has not

7   been changed since the writable vdisk was created;

8   a processor that searches each field of the writable vdisk for a hole; and

9   the processor that fills each hole in the writable vdisk so that each hole in the

10   writable vdisk points to ~~the~~ a data block referenced by ~~the~~ a corresponding backing store

11   indirect block to fill each hole of the ~~writeable~~ writable vdisk with the data block

12   referenced by the corresponding backing store indirect block and thus update the writable

13   vdisk to reference both the data which is unchanged since the writable vdisk was created

14   and the data which has been changed since the writable vdisk was created.

1   32. (Currently Amended) The system of claim 31, ~~further comprising:~~ wherein the

2   processor further dirties the data block pointed to by the backing store indirect block ~~are~~

3   ~~dirtied~~ to enable write allocation of the dirty data block without altering a data content of

4   the data block.

1   33. (Currently Amended) The system of claim ~~31~~ 32 ~~further comprising:~~ wherein the

2   processor further:

3   chooses a new pointer ~~chosen~~ for a newly allocated data block containing an

4   unaltered data content;

5   sets bits ~~are set~~ in ~~a~~ block allocation structures for the newly allocated data block;

6   and

7   places a new pointer to the newly allocated data block placed into a field of the

8   writable vdisk indirect block to replace the hole.

1   34. (Currently Amended) The system of claim 33, further comprisingwherein the

2   processor further:

3   frees the dirty data block is freed; and

4   writes the newly allocated data block is written to disk.

1   35. (Currently Amended) The system of claim 34 further comprising:wherein the

2   processor further releases an association of the writable vdisk to the backing store is

3   released to thereby separate the writable vdisk data blocks from the backing store data

4   blocks.

1   36. (Currently Amended) The system of claim 31, further comprising:wherein the

2   processor further has logical volume block numbers (VBNs) included in the pointers

3   contained in the a writable vdisk indirect block fields of the writable vdisk and the a

4   backing store indirect block fields of the backing store.

1   37. (Currently Amended) The system of claim 3136, further comprising: wherein the

2   processor further uses a zero logical volume block number (VBN) used as the an invalid

3   pointers in the hole contained in the writable vdisk indirect block fields.

1   38. (Currently Amended) The system of claim 31, further comprising:wherein the

2   processor further uses a writable vdisk level 1 buffer used for the a plurality of fields in

3   the a writable vdisk indirect block of the writable vdisk and use a backing store level 1

4   buffer used for the a plurality of fields in the backing store indirect block.

1   39. (Currently Amended) A non-transitory computer readable medium containing

2   executable program instructions executed by a processor comprising:

3   program instructions that create a writable virtual disk (vdisk) at a selected time,

4   the writable vdisk referencing changes in data stored in the data storage system after the

5   writable vdisk was created, the writable vdisk having a plurality of holes where each hole

6   instructs the storage system to examine a corresponding virtual block number pointer in a

7   backing store;

8   program instructions that maintain the backing store, the backing store

9   referencing data stored in the data storage system which has not been changed since the

10   writable vdisk was created;

11   program instructions that search each field of the writable vdisk for a hole; and

12   program instructions that fill each hole in the writable vdisk to point to ~~the~~ a data

13   block referenced by ~~the~~ a corresponding backing store indirect block to fill each hole of

14   the ~~writeable~~ writable vdisk with the data block <u>referenced by the corresponding backing</u>

15   <u>store indirect block</u> and thus update the writable vdisk to reference both the data which is

16   unchanged since the writable vdisk was created and the data which has been changed

17   since the writable vdisk was created.

1   40. (Currently Amended) A method for operating a data storage system, comprising:

2   creating a writable virtual disk (vdisk) at a selected time, the writable vdisk

3   referencing changes in data stored in the data storage system after the writable vdisk was

4   created, the writable vdisk having a plurality of holes where each hole instructs the data

5   storage system to examine a corresponding virtual block number pointer in a backing

6   store;

7   maintaining the backing store, the backing store referencing the data stored in the

8   data storage system which has not been changed since the writable vdisk was created;

9   searching, by a background task process, each field of the writable vdisk for a

10   hole;

11   for each hole in the ~~writeable~~ writable vdisk, marking as dirty ~~the~~ a corresponding

12   data block pointed to by ~~the~~ a backing store indirect block without modifying the

13   corresponding data block; and

14   performing a write allocation to replace each hole in the writable vdisk to point to

15   the data block marked as dirty and referenced by the corresponding backing store indirect

16   block to update the writable vdisk to reference both the data which is unchanged since the

17   writable vdisk was created and the data which has been changed since the writable vdisk

18   was created.

1   41. (Currently Amended) A data storage system, comprising:

2   a writable virtual disk (vdisk) created at a selected time, the writable vdisk

3   referencing changes in data stored in the data storage system after the writable vdisk was

4   created, the writable vdisk having a plurality of holes where each hole instructs the data

5   storage system to examine a corresponding virtual block number pointer in the backing

6   store;

7   the backing store referencing the data stored in the data storage system which has

8   not been changed since the writable vdisk was created;

9   a background task processor that searches each field of the writable vdisk for a

10   hole; and

11   the background task processor that marks as dirty, for each hole in the ~~writeable~~

12   writable vdisk, ~~the~~ a corresponding data block pointed to by ~~the~~ a backing store indirect

13   block without modifying the corresponding data block, and performs a write allocation to

14   replace each hole in the writable vdisk to point to the data block marked as dirty and

15   referenced by the corresponding backing store indirect block to update the writable vdisk

16   to reference both the data which is unchanged since the writable vdisk was created and

17   the data which has been changed since the writable vdisk was created.

1   42. (Currently Amended) A non-transitory computer readable medium containing

2   executable program instructions executed by a processor, comprising:

3   program instructions that create a writable virtual disk (vdisk) at a selected time,

4   the writable vdisk referencing changes in data stored in the data storage system after the

5   writable vdisk was created, the writable vdisk having a plurality of holes where each hole

6   instructs the data storage system to examine a corresponding virtual block number pointer

7   in a backing store;

8   program instructions that maintain the backing store, the backing store

9   referencing the data stored in the data storage system which has not been changed since

10 the writable vdisk was created;

11   program instructions that search, by a background task process, each field of the

12 writable vdisk for a hole;

13   program instructions that mark as dirty, for each hole in the ~~writeable~~ writable

14 vdisk, ~~the~~ a corresponding data block pointed to by ~~the~~ a backing store indirect block

15 without modifying the corresponding data block; and

16   program instructions that perform a write allocation to replace each hole in the

17 writable vdisk to point to the data block marked as dirty and referenced by the

18 corresponding backing store indirect block to update the writable vdisk to reference both

19 the data which is unchanged since the writable vdisk was created and the data which has

20 been changed since the writable vdisk was created.